An Adaptive Multipopulation Framework for Locating and Tracking Multiple Optima

Changhe Li, *Member, IEEE*, Trung Thanh Nguyen, Ming Yang, Michalis Mavrovouniotis, *Member, IEEE*, and Shengxiang Yang, *Senior Member, IEEE*

***Abstract*—Multipopulation methods are effective in solving dynamic optimization problems. However, to efficiently track multiple optima, algorithm designers need to address a key issue: how to adapt the number of populations. In this paper, an adaptive multipopulation framework is proposed to address this issue. A database is designed to collect heuristic information of algorithm behavior changes. The number of populations is adjusted according to statistical information related to the current evolving status in the database and a heuristic value. Several other techniques are also introduced, including a heuristic clustering method, a population exclusion scheme, a population hibernation scheme, two movement schemes, and a peak hiding method. The particle swarm optimization and differential evolution algorithms are implemented into the framework, respectively. A set of multipopulation-based algorithms are chosen to compare with the proposed algorithms on the moving peaks benchmark using four different performance measures. The effect of the components of the framework is also investigated based on a set of multimodal problems in static environments. Experimental results show that the proposed algorithms outperform the other algorithms in most scenarios.**

***Index Terms*—Dynamic optimization, multimodal optimization, multipopulation optimization, population adaptation.**

I. Introduction

GENERALLY speaking, multipopulation methods (MPMs) use more than one population to cooperatively search in different areas in the fitness landscape to locate multiple optima or the global optimum. They are widely used to track multiple optima/peaks in parallel for dynamic optimization problems (DOPs). The motivation is that each population covers a different peak so that tracking the global optimum would be easy if the global optimum moves to the area where a population is covering. For a DOP with a certain number of peaks, tracking the global optimum would be inefficient if the number of populations is far less than the number of peaks; on the other hand, the tracking would also be inefficient if the number of populations is far more than the number of peaks. Therefore, to efficiently solve DOPs by MPMs, one key issue is to adapt the number of populations [20], [27].

A good practice is to choose the number of populations in relation to the number of optima, if the number of optima is known [19]. Many experiments have shown that an inappropriate number of populations would negatively affect the performance of MPMs [3], [6], [19], [20], [37]. This problem becomes more challenging for DOPs with a changing number of optima/peaks [20], [37]. In the literature of MPMs for DOPs, many studies focus on a fixed number of populations [6], [14], [22]. Although algorithms based on a dynamic number of populations were proposed [9], [18], [29], the total number of individuals is fixed. This limitation constrains the adaptability of such algorithms. For example, it would be impossible for those algorithms to track all optima in parallel when the number of optima is more than the total number of individuals. To the best of our knowledge, only three versions of adaptive MPMs [3], [20], [37] have been proposed for DOPs so far. The difficulties in developing such algorithms lie in the following factors: 1) The number of optima in the fitness landscape is unknown and 2) The relationship between the right number of populations and the number of optima is also unknown, even if *a priori* knowledge of the number of optima is available.

To address the aforementioned issues for MPMs, this paper proposes an adaptive multipopulation (AMP) framework based on an adaptive mechanism. The adaptive mechanism learns from the change in the number of populations by means of interacting with environments and, in turn, guides the change toward a promising direction. To implement the adaptive mechanism in an efficient way, a database is used to record the changes. By doing so, the AMP framework is able to predict the number of populations to be adjusted based on historical data stored in the database. This is the most important difference between this paper and existing studies [3], [20], [37], wherein the adaptation is based only on the knowledge of the current evolving status. (See more discussions in Sections II and III-B later.)

To make the AMP framework work, two fundamental components are developed. One is a heuristic clustering method,

Manuscript received May 16, 2015; revised August 18, 2015; accepted November 18, 2015. This work was supported in part by the National Natural Science Foundation of China under Grant 61203306 and Grant 61305086, in part by the Engineering and Physical Sciences Research Council of U.K. under Grant EP/K001310/1, in part by the British Council UK-ASEAN Knowledge Partnership Grant, and in part by the British Council Newton Institutional Links Grant. *(Corresponding author: Ming Yang.)*

C. Li and M. Yang are with the Hubei Key Laboratory of Intelligent Geo-Information Processing, China University of Geosciences, Wuhan 430074, China (e-mail: changhe.lw@gmail.com; yangming0702@gmail.com).

T. T. Nguyen is with the School of Engineering, Technology and Maritime Operations, Liverpool John Moores University, Liverpool L3 3AF, U.K. (e-mail: t.t.nguyen@ljmu.ac.uk).

M. Mavrovouniotis and S. Yang are with the School of Computer Science and Informatics, De Montfort University, Leicester LE1 9BH, U.K. (e-mail: mmavrovouniotis@dmu.ac.uk; syang@dmu.ac.uk).

Digital Object Identifier 10.1109/TEVC.2015.2504383

which clusters a certain number of random individuals to a set of populations without overlapping areas between each other. The other is a population exclusion method, which is responsible for removing overlapping populations during the optimization process so that no more than one population would converge on the same peak. To enhance the performance of the AMP framework, several other components are introduced: a peak hiding scheme, a population hibernation scheme, and two movement schemes for the best individuals. The peak hiding scheme is used to hide peaks that have been explored so that no more populations would move to explored peaks. The population hibernation scheme is used to save function evaluations. The two movement schemes for the best individuals are Brownian and Cauchy movements, respectively, where the former helps a converging population converge on a peak's summit rather than on its slope and the latter helps to transform stagnating populations to converging populations.

The rest of this paper is organized as follows. Section II reviews related research in the literature of MPMs for DOPs. The components of the proposed AMP framework and two instantiated algorithms are introduced in detail in Section III. Experimental studies are provided in Section IV. Finally, Section V concludes this paper.

II. Related Work

One early version of MPM is the self-organizing scouts (SOSs) algorithm [9] proposed to solve the moving peaks benchmark (MPB). SOS starts from a parent population that explores new promising peaks. Child populations, which are used to track peaks, are generated by splitting off from the parent population when a certain condition is satisfied (i.e., forking generations are detected). Based on the scout model of SOS, several other algorithms were proposed. A multiswarm algorithm based on the particle swarm optimization (PSO) was developed in [4], where a part of swarms exploits peaks that have been detected, and the remaining swarms keep exploring new peaks. Another multiswarm forking algorithm [42] was developed to solve the MPB by applying the same idea with SOS to PSO. A fast multiswarm optimization algorithm was proposed in [17] by using a similar idea with SOS to organize multiple swarms, except that child swarms are created when changes are detected. To give more computing time to productive swarms than unproductive swarms, a hibernation multiswarm optimization (HmSO) was proposed in [14]. A child swarm is forced to hibernate if its radius is less than a converging threshold value and the fitness of its best particle is worse than the fitness of the global best particle by a predefined level.

Instead of the splitting idea, many algorithms use a regrouping idea to create populations. A popular algorithm is the speciation-based PSO (SPSO) [29]. As in SOS, SPSO also starts with a large size swarm. Differently, the initial swarm is divided into a number of subswarms by a speciation-based rule. A swarm is created by combining the best particle with particles that are close to that best particle (i.e., the distance to the best particle is less than a predefined value). Once a swarm is constructed, its particles are removed from the initial swarm. This procedure is repeated until the initial swarm is empty. The construction of swarms is performed every iteration. Based on this regrouping idea, many improved versions of SPSO and variants were proposed.

A mechanism to remove duplicated particles was introduced to enhance the performance of SPSO [30]. Another improved version of SPSO (rSPSO) was developed by integrating a least square regression method [2]. Recently, a multipopulation harmony search algorithm was proposed [39], whereby a harmony search method [12] is used for each population to locate peaks. The best individuals of converged populations are kept to replace redundant ones.

In addition to the speciation-based approaches, niching techniques are also widely used to maintain multiple populations. An adaptive niching PSO was proposed in [1] whereby niching radii can be calculated adaptively. A vector-based PSO (VBPSO) was developed in [34], in which the dot product of two vectors is used to identify niches. The algorithm shows a competitive performance for multimodal optimization. Thereafter, VBPSO was extended in [35] for DOPs. A cluster-based differential evolution (DE) algorithm for niching was proposed in [26], wherein different kinds of strategies were introduced to efficiently track multiple peaks. Recently, historical information was used to create niches without additional evaluations in [21].

Instead of creating populations during the runtime, many MPMs start with a fixed number of populations. One of the most popular algorithms is the atomic swarm model [5] where three kinds of particles with different roles are defined. They are charged particles, quantum particles, and neutral particles. In each swarm in the model, either charged particles (mCPSO [6]) or quantum particles (mQSO [6]) play the role of maintaining diversity, and neutral particles are used to locate peaks. An exclusion principle ensures that only one swarm surrounds a single peak and an anticonvergence principle is introduced to explore new promising peaks.

Motivated by the atomic model [5], several similar algorithms have been proposed. A multipopulation dynamic DE (DynDE) [25] algorithm was proposed, wherein four types of individuals, named DE, entropy DE, quantum and Brownian, are defined. An improved version of mQSO was proposed in [11], where two heuristic rules are applied to further enhance the diversity when changes occur. A fuzzy-C-means strategy was introduced to adapt the exclusion radius in [33]. In the algorithm, all particles are transformed to quantum particles till the next iteration when a change is detected. Recently, a cooperative quantum PSO [40] was proposed by using a cooperative framework introduced in [7]. A multiswarm algorithm, called finder-tracker multiswarm PSO (FTMPSO), was proposed in [46] by integrating several schemes, including a finder scheme, a tracker scheme, a change detection scheme, a wakening and sleeping scheme, and a local search scheme. Thereafter, Yazdani *et al.* [45] proposed a multiswarm algorithm based on a new artificial fish swarm algorithm (mNAFSA). In the algorithm, a mechanism of finding and covering potential optimum peaks was proposed.

Instead of using different types of individuals in each population, several MPMs use different search algorithms for different populations. A collaborative evolutionary swarm optimization (CESO) algorithm was proposed in [22], whereby two swarms, using the crowding DE [38] and PSO, respectively, cooperate with each other using a collaborative principle to track the global optimum. Thereafter, a new version of CESO was proposed in [23], called evolutionary swarm cooperative algorithm, where another swarm based on PSO was added, and the cooperation principle was updated. A multienvironmental cooperative model [15] was introduced to deal with DOPs that have different subproblems or environments.

Another important kind of MPMs are clustering-based algorithms. A popular example is the clustering PSO (CPSO) [44], where a hierarchical clustering method is used to create multiple swarms by clustering a random swarm whenever a change is detected. In CPSO, an overlapping detection principle was proposed to identify whether two swarms crowd around one single peak when they move into each other's search area. One of the two swarms is removed if they are not overlapped but overcrowded (i.e., they crowd around one single peak). Thereafter, CPSO was enhanced to a version, called CPSOR, which does not need to detect changes. CPSOR introduces a principle that the population diversity is automatically increased once the total number of individuals is less than a threshold value. Recently, a new cluster-based DE algorithm was proposed in [13] in which k-means is used to create populations whenever a change is detected. The number of populations may vary after every certain time span, depending on the performance of the algorithm.

All methods mentioned above do not address one important issue of MPMs for DOPs, which is how to adapt the number of populations to dynamic environments, especially in the situation in which the number of peaks changes over time. Several attempts have been made to address this difficult issue. A self-adaptive multiswarm optimizer (SAMO) [3] was developed based on the mQSO [6]. SAMO starts with a single free swarm. The number of free swarms is decreased when some of them are converging. SAMO creates a new free swarm if there is no free swarm. Converging swarms are identified by simply checking their radius against a predefined value. This way, the number of populations will be adaptively adjusted. Accordingly, the search area of each swarm is also adjusted by a formula that takes the number of populations into account. The motivation of SAMO was adopted in a DE algorithm, called DynPopDE [37]. A new free population is created when one population is identified as a stagnating population. A stagnating population will be removed if it is identified for reinitialization due to exclusion.

Note that, in this paper a population is considered "converging" if the average distance among individuals begins to decrease. A population is considered "converged" if the average distance among individuals is less than a threshold value. A population is considered "stagnating" if it stops improving permanently but does not show any trend of converging [16].

Recently, an adaptive multiswarm optimizer (AMSO) [20] was proposed. AMSO maintains a number of populations obtained by the clustering method used in [44]. Due to the overlapping handling principle [44], the number of populations will decrease after each diversity increasing point. A certain number of individuals are introduced when the drop rate of the number of populations over a time span is less than a small value. The number of individuals to be adjusted depends on the difference of the number of populations between the current increasing point and the previous increasing point. This way, AMSO is able to adaptively adjust the number of populations during the runtime.

Although there are three adaptive MPMs for DOPs, the principles to adjust the number of populations simply rely on the current information available, e.g., no free population in SAMO [3], the appearance of stagnating populations in DynPopDE [37], and the difference of the number of populations at the current and previous increasing points in AMSO [20]. Simply relying on the current information to adjust the number of populations may lead to mistakes due to the complex nature of dynamic environments. In order to address the aforementioned issues of adapting the number of populations, we introduce the AMP framework in this paper, which is described below.

III. Adaptive Multipopulation Framework

The proposed AMP framework consists of three major components, such as clustering, tracking, and adapting. The clustering component is used to create a number of populations, which have no overlapping search areas with each other. The tracking component allows locating and tracking peaks using any single-population-based search algorithm. Finally, the adapting component, which is the key component of the AMP framework, adjusts the total number of populations by predicting what will be the best number of populations.

To enhance the adapting component, the AMP framework integrates several other components: a population exclusion scheme, a hibernation scheme, a peak hiding scheme, and two movement schemes for the best individuals. The detailed description for each component is given in the following sections.

A. Heuristic Clustering

A single linkage hierarchical clustering method [24] is used to create nonoverlapping populations. The following notations are given.

1) $d(i, j)$ is the Euclidean distance between two individuals i and j in the D-dimensional space.
2) $D(C_1, C_2) = \min_{i \in C_1, j \in C_2} d(i, j)$ is the distance between two clusters C_1 and C_2.
3) $R(C) = (1/|C|) \sum_{i \in C} d(i, i^*)$ is the radius of C, where i^* is the centroid of C, i.e., position $\vec{x}_{i^*} = \sum \vec{x}_i / |C|$.
4) $d_{\text{inter}} = \sum_{C_1, C_2 \in \mathbb{C}} D(C_1, C_2)$ is the sum of intercluster distances between each pair of clusters in a list $\mathbb{C}$.
5) $d_{\text{intra}} = \sum_k \sum_{i,j \in C_k} d(i, j)$, $k = 1, 2, \ldots, |\mathbb{C}|$ is the sum of intracluster distances of all clusters in $\mathbb{C}$, where the intracluster distance of cluster C_k is the sum of all the distances between each pair of individuals in C_k.

Algorithm 1 Cluster(C')

1: **for** $i < |C'|$ **do** $C_i \leftarrow C'[i]$; **end for** ▷ C' is a population.
2: Calculate d_{inter} and d_{intra} of $\mathbb{C}$; ▷ $\mathbb{C}$=[C_1, C_2, ...]
3: **while** $d_{intra} < d_{inter}$ **do**
4: Merge C_i and C_j, where $D(C_i, C_j)$=min$_{i \neq j < |\mathbb{C}|} D(C_i, C_j)$;
5: Update d_{inter} and d_{intra};
6: **end while**
7: Return $\mathbb{C}$;

Algorithm 2 Adjust( )

1: $p \leftarrow |N_t^e - N_{t-1}^e|/M$; ▷ p is a probability of taking an action.
2: **if** $p >= 1$ **then** $f \leftarrow 1$;
3: **else if** $p = 0$ **then** $f \leftarrow 0$;
4: **else** ▷ *rand*() returns a random number in [0,1].
5: **if** *rand*() $< p$ **then** $f \leftarrow 1$; **else** $f \leftarrow 0$; **end if**
6: **end if**
7: $I_{t+1}^b \leftarrow N(\mathbb{D}_\mu[N_t^e], \mathbb{D}_\sigma[N_t^e]) + 5f \cdot (N_t^e - N_{t-1}^e)$;

Algorithm 1 shows the workflow of the clustering method to cluster a population C'. It first creates a list $\mathbb{C}$ of clusters, with each cluster containing only a single individual. Then, for each iteration, it merges a pair of clusters that have the smallest distance among all pairs of clusters in $\mathbb{C}$ and satisfies the condition $d_{\text{intra}} < d_{\text{inter}}$. When d_{intra} is equal to or greater than d_{inter}, the clustering procedure terminates. Then, all clusters in $\mathbb{C}$ are appended to a population list $\mathbb{P}$, which is empty initially. The benefit of this heuristic clustering method, compared with the existing method used in [19], [20], and [44], is that it does not need to manually tune parameters, such as the lower and upper bounds for the cluster size. Note that this method cannot guarantee that the sizes of all obtained clusters meet a required minimum population size for an algorithm (e.g., in this paper the minimum population size is five for the DE and two for PSO). In this paper, such clusters are ignored, but their individuals are used as random individuals when a new random population (C') is created. (See step 20 of the framework in Algorithm 5, which will be introduced later.)

B. Adapting Populations

There are two main concerns regarding the adaptation of the number of populations to changes: 1) when to make an adjustment and 2) how many populations to be adjusted. For the first concern, many researchers consider the moment when a change occurs as the time to make an adjustment (e.g., increasing/introducing diversity or reusing information learned from the past in many studies reviewed above [6], [8], [18], [22], [23], [30], [44]). However, this strategy has a limitation: It may not work if changes are hard or impossible to detect by reevaluating a set of points in complex environments, such as dynamic environments where a part of the fitness landscape changes [20], [27] or there is noise.

The adaptive MPM algorithms in [3], [20], and [37] do not use the above strategy. However, these algorithms have other issues. In SAMO [3], the issue is that a stagnating swarm with a large radius will be incorrectly regarded as a converging swarm if the total number of swarms is significantly less than the number of peaks. As a result, the stagnating swarm will not be eliminated and consequently no new swarm will be created. Although SAMO may still work in this case because stagnating swarms will normally transform to converging swarms after a change occurs, this situation may affect the performance of SAMO. In DynPopDE [37], this stagnating issue is taken into account, where a population is considered as a stagnating population if its best individual does not improve over two successive iterations. However, this way of identifying stagnating populations is not very effective, and it may cause the generation of too many populations. In AMSO [20], new populations are added if the number of populations over a time span is dropped beyond a threshold. However, this requires setting the correct values for the time span and the drop threshold, which may not be an easy task [20].

Based on the above considerations, we aim to find a simple yet effective way to identify the moment when populations are converging. To achieve this aim, the average radius ($r_{\text{avg}}^{\text{conv}}$) of nonstagnating populations is monitored. Population adjustments are triggered when $r_{\text{avg}}^{\text{conv}}$ is less than a threshold value of $\theta \cdot S$ (S is the distance between two farthest points in the solution space). Stagnating populations should be excluded because they can seriously affect the judgement of whether nonstagnating populations are converging if they have large search radii. In this paper, a population C is regarded as a stagnating population if the following three conditions are satisfied: 1) Its radius does not shrink after a certain number of successive iterations, which is equal to $|C|$ [43]; 2) Its radius is greater than the average radius of all populations in $\mathbb{P}$; and 3) Its radius is greater than $\theta \cdot S$. Note that this method does not guarantee that a population, which satisfies these conditions, is a real stagnating population. However, a real stagnating population would satisfy these conditions.

The second concern is very difficult to address directly due to the two difficulties mentioned in Section I, namely the lack of knowledge on the number of optima and on the correlation between the number of optima and the number of populations. However, we may still be able to indirectly address this concern by observing the change in the algorithm behavior. For example, it can be easily inferred that in an MPM algorithm, the number of survived populations (i.e., populations that find new, unexplored peaks—these populations will survive the exclusion procedure in MPMs) is proportional with the number of peaks in the fitness landscape. If the number of peaks increases, so does the number of survived populations, and vice versa. Therefore, we can use this heuristic relationship between the number of survived populations and the number of peaks to predict the number of populations to be adjusted.

Given this heuristic, we are able to address the second concern by a two-step process. The first step is to decide whether to increase, decrease, or make no change to the total number of individuals. To adjust the number of populations, in the AMP framework, we have to first adjust the number of individuals, since populations are obtained only by clustering individuals. A map (to be introduced later) is defined to represent the relationship between the number of populations at the end of an evolving phase and the number of individuals at the beginning of the evolving phase. To determine the appropriate

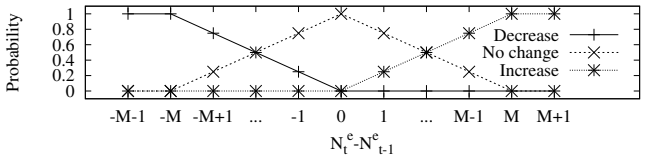

Fig. 1. Probability of increasing, decreasing, or making no change to the number of individuals, where $[-M, M]$ is the probabilistic range and N_t^e is the number of populations at the end of evolving phase t.

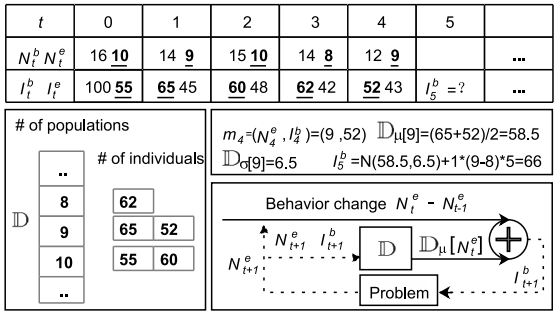

Fig. 2. Illustration of the adaptation in the AMP framework, where N_t^b and N_t^e(I_t^b and I_t^e) are the number of populations(individuals) at the beginning and the end of evolving phase t, respectively, $D_\mu[N_t^e]$ and $D_\sigma[N_t^e]$ are the mean and standard deviation, respectively, of individuals with map items that have N_t^e populations.

action toward the number of individuals (increasing, decreasing, or making no change), a probabilistic prediction scheme, as shown in Algorithm 2 and illustrated in Fig. 1, is introduced, where N_t^e is the number of populations at the end of evolving phase t.

In this scheme, one of the three actions is taken depending on a probability, which is obtained by $|N_t^e - N_{t-1}^e|/M$, and the sign of $N_t^e - N_{t-1}^e$. For example, it will increase the number of populations if the return value of Algorithm 2 is $f = 1$ and $N_t^e - N_{t-1}^e$ is greater than zero. The value of $M > 0$ determines the probabilistic range (the scheme is deterministic if M is one). As illustrated in Fig. 1, the greater the difference between N_t^e and N_{t-1}^e, the greater the probability of increasing/decreasing the number of individuals.

The second step is to estimate how many individuals should be set in the near future. Different from the existing methods [3], [20], [37], where the number of populations changes based on the current evolving status, in this paper, the adaptation is achieved based on historical data and a feedback value. To achieve this objective, we first create a map item (m_t) from the current number of populations (N_t^e) to the total number of individuals (I_t^b) at the beginning of the current phase (t) whenever a new adjustment is triggered. Note that for the first map item, an overly large initial population (far larger than needed, e.g., 500 individuals for the 10-peak MPB in the 2-D space) would affect the estimation if we use I_0^b (the initial population size) to create it. To address this issue, we use I_0^e (the number of individuals at the end of phase $t = 0$) for the first map item instead of I_0^b. Then, the map item is added into a database $\mathbb{D}$, which records all map items created since the start of the run. To estimate the number of individuals, we average the number of individuals of all map items that have the same number of populations as the current map item. Then, the number of individuals for the next evolving phase is estimated by

$$I_{t+1}^b \leftarrow N\left(\mathbb{D}_\mu\left[N_t^e\right], \mathbb{D}_\sigma\left[N_t^e\right]\right) + 5f \cdot \left(N_t^e - N_{t-1}^e\right) \qquad (1)$$

where $N(a, b)$ is a normally distributed random number with the mean a and the standard variation b, $D_\mu[N_t^e]$ and $D_\sigma[N_t^e]$ are the mean and standard deviation, respectively, of individuals with map items that have N_t^e populations, and $f \in \{0, 1\}$ denotes whether to take the feedback value of $N_t^e - N_{t-1}^e$ into account.

Finally, a random population C' is created with a size of $I_{t+1}^b - I_t^e$ (see step 20 in Algorithm 5). Note that I_{t+1}^b needs to be repaired in the case of $I_{t+1}^b <= I_t^e$, where C' is created with ten individuals to guarantee that the diversity is increased when all nonstagnating populations are converging. In this situation, the size of C' should be small as many overcrowding populations would be created if a large value is used. Although the exclusion scheme (to be introduced in Section III-C later) is able to remove overcrowding populations, such populations still waste computing resources. Preliminary experimental results show that a value of ten is a reasonable choice.

Fig. 2 shows an example of the estimation process with the database containing five map items. In Fig. 2, the number of populations at the current adjustment point is nine and the current map item is $m_4 = (9, 52)$. Then, the number of individuals for the next evolving phase (I_5^b) can be obtained by (1), which is 66. As shown in the bottom right graph in Fig. 2, the adaptation of the AMP framework is a feedback loop. The number of populations changes over time by interacting with the environment. For example, if increasing the number of populations makes the AMP mechanism find new peaks, the AMP mechanism will continuously take this action until no new peaks can be found. On the other hand, if the number of peaks decreases, the number of populations will likely decrease accordingly. The database keeps receiving feedbacks of the algorithm behavioral changes, which further guides the future changes in the algorithm behavior. Therefore, the AMP mechanism is a self-regulating framework.

Note that the constant value of 5 in (1) is a step size. Its value should be roughly equal to the average population size. In this way, the number of populations for the next evolving phase would be close to the expected value. An overly large/small value will cause too many/few populations to be generated. Considering the average population size for all instances tested in this paper (4.687) and the minimum population size for the DE (5) and PSO (2) algorithms, a value of 5 is chosen for this constant.

C. Population Exclusion

Overcrowding populations, which are several populations that surround the same peak, are not allowed in the AMP framework. In this paper, two populations are detected as overcrowding if both have at least one individual in each other's search areas. Then, the population with the worst best individual is removed from $\mathbb{P}$ (see step 15 in Algorithm 5). Note that this method cannot guarantee that two populations, which are

detected as overcrowding populations, really search on a same peak. However, it guarantees that a peak, which is already in the search area of one population, cannot be taken by any other population.

Although this scheme is simple, it plays an important role in the AMP framework. It has two functions. First, it saves evaluations due to the removal of redundant/overcrowding populations. Second, it enables converging populations to distribute on different peaks, thus the difference of the number of populations at the end of two successive evolving phases [i.e., the second part in (1)] is valid as each explored peak corresponds to a unique population.

D. Avoidance of Explored Peaks

Exploring peaks that have already been explored wastes computational resources and hence deteriorates an algorithm's performance. This is an important issue for evolutionary algorithms (EAs) [28]. To address this issue, we propose a peak hiding technique in this paper. A peak is assumed to be explored when there exists a population where the distance between its best individual and the peak is less than a value (ϵ_s), and the difference between the objective value of the best individual and the peak height is also less than a small value (ϵ_o) in the case that the peak location is known. Otherwise, a peak is regarded as an explored peak only if a population's radius shrinks to a small value of $1E$-9, and the location where the population converges is assumed to be the location of the peak. It is important to note that this assumption does not hold if the population converges on a peak's slope rather than on its summit. (See Section III-F for one possible solution introduced later.)

To avoid populations researching peaks that have been explored, an idea is to remove the attraction of explored peaks. The idea works as follows. All peaks that have been explored are kept. For each explored peak, a set of vectors from the peak location to the boundary of its basin of attraction will be created as necessary. Initially, the set is empty. For an individual i, we find the closest explored peak p. In the vector set with peak p, if there does not exist a vector v' that makes its angle to $\vec{x}_i - \vec{x}_p$ less than three degrees, then a new vector v is created from p in the direction of $\vec{x}_i - \vec{x}_p$. The length of v gradually increases by a small step ($0.1\epsilon_s$) until a turning or a boundary point is found. Then v is added to the vector set of p. The fitness value of i is set to the fitness value of the worst solution found so far if $|\vec{x}_i - \vec{x}_p|$ is less than $|v|$ or $|v'|$, depending on whether the condition above is met. It is worth noting that it is possible to set the values of the angle threshold and the step to be smaller than the default values so that the vector v can be more precise. But by doing so, more evaluations will be needed. Our experimental studies show that the two default values are small enough for all the test problems in this paper.

Fig. 3 illustrates the procedures of finding such vectors ($v1$ and $v2$) for an explored peak p in a 1-D problem $f(x)$. In the figure, the triangle point is a turning point with peak p, and the star point is a boundary point. The fitness of individual 2 will not be degraded as $|x_2 - x_p|$ is greater than $|v2|$. However, the fitness of individual 1 will be set to the fitness of the worst individual found so far as $|x_1 - x_p|$ is less than $|v1|$. In this way, all individuals that fall in the basin of attraction of peak p will not move toward p any more since they will get the worst fitness by doing so.

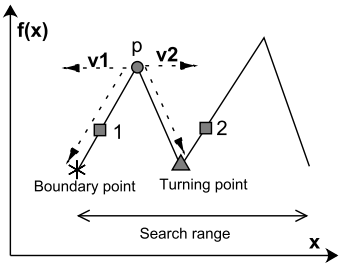

Fig. 3. Illustration of the peak hiding method.

The peak hiding scheme will cause an issue for a newly generated population if it is placed in the basin of attraction of an explored peak. The issue is that such a population is likely to become a stagnating population, as all of its individuals have the same fitness (i.e., the fitness of the worst individual) as long as all of them remain in the basin of attraction. (See one possible solution later in Section III-F.)

E. Population Hibernation and Wakening

To save function evaluations, inspired by [14], a population will hibernate once it finds a peak and will wake up when the diversity adjustment is performed. Hibernating populations will not evolve until they wake up. It is important to note that waking up hibernating populations (step 17 in Algorithm 5) is necessary when the AMP mechanism is running in dynamic environments. This is because when a change occurs, those hibernating populations will have outdated memories, and they must wake up in time to relocate peaks that have moved. In the AMP framework, we choose the diversity adjustment point as the time point to wake up hibernating populations.

F. Movements for the Best Individual

For a nonstagnating population C, in order to quickly track a moving peak or a better peak not covered by any population, the best individual $\vec{x}_{\text{best}}$ performs a Brownian movement [25] within the search area of C at each iteration

$$\vec{x'}_{\text{best}} = N(\vec{x}_{\text{best}}, R(C)) \tag{2}$$

and it will be replaced if a better solution $\vec{x'}_{\text{best}}$ is found.

However, if C is a stagnating population, it may not benefit from the Brownian movement. However, helping such a population to jump out of its search area could help transform the population to a converging population. To achieve this, we allow the population's best individual to perform a Cauchy movement at each iteration

$$\vec{x'}_{\text{best}} = Q(\vec{x}_{\text{best}}, S/2) \tag{3}$$

where $Q(a, b)$ is a random number of Cauchy distribution with location parameter a and scale parameter b.

The Brownian movement is important for a population that (nearly) converges on a peak's slope. In our experimental studies, we observe that a population sometimes converges at some

Algorithm 3 PSO

1: **for** each particle i **do**
2: Update particle i according to Eqs. (4a) and (4b);
3: **if** $f(\vec{x}_i) < f(\vec{x}_{p_i})$ **then** $\vec{x}_{p_i} := \vec{x}_i$; **end if**
4: **if** $f(\vec{x}_i) < f(\vec{x}_g)$ **then** $\vec{x}_g := \vec{x}_i$; **end if**
5: **end for**

point on the slope of a peak rather than on its summit due to the lack of diversity and that more than one population converges on the slope of the same peak (this will seriously affect the performance of the AMP mechanism, see the comparison results in Table VII in Section IV-D4). The Brownian movement is able to prevent such populations from converging on the slope of a peak. This is because the Brownian movement will pull the best individual to a higher point nearby until the whole population reaches the peak's summit.

The Cauchy movement is particularly helpful for a newly generated population if it is placed in the basin of attraction of an explored peak. Such a population is likely to become a stagnating population, as all its individuals have the same fitness due to the peak hiding scheme. The population cannot get improved if it remains in the basin of attraction of the explored peak. The long jump of the Cauchy movement gives the population a larger probability of jumping out of the trapped area.

G. Instantiation of the AMP Framework

In the AMP framework, any single-population-based algorithm can be applied to the search task, which is performed at step 6 in Algorithm 5, without any modification. Although any population-based algorithm can serve the search task, an algorithm that has a good local search and fast convergence capability is preferred. This is because it is more appropriate for a population to search within its local area only. In this way, the AMP mechanism will be able to track multiple good peaks. In this paper, the AMP framework will be instantiated with a PSO algorithm and a DE algorithm, respectively, with the suggested features.

1) AMP With PSO: The PSO with an inertia weight [36] is used in this paper. The velocity and position of particle i are updated as

$$\vec{v'}_i = \omega\vec{v}_i + \eta_1\vec{r}_1\left(\vec{x}_{p_i} - \vec{x}_i\right) + \eta_2\vec{r}_2\left(\vec{x}_g - \vec{x}_i\right) \tag{4a}$$

$$\vec{x'}_i = \vec{x}_i + \vec{v'}_i \tag{4b}$$

where $\vec{x'}_i$ and $\vec{x}_i$ represent the current and previous positions of particle i, respectively; $\vec{v'}_i$ and $\vec{v}_i$ are the current and previous velocities of particle i, respectively; $\vec{x}_{p_i}$ and $\vec{x}_g$ are the best positions found by particle i so far and found by the whole swarm so far, respectively; $\omega = 0.7298$ and $\eta_1 = \eta_2 = 1.496$ are constant parameters, whose values were suggested by [41]; and $\vec{r}_1$ and $\vec{r}_2$ are vectors of random numbers uniformly generated within $[0.0, 1.0]$ for each dimension. In the PSO, the information of the best particle is shared with all the other particles, the whole swarm will quickly converge at the location of the best particle. Algorithm 3 presents the framework of the PSO for minimization problems.

Algorithm 4 DE

1: **for** each individual i **do**
2: Generate a donor vector $\vec{v}$ by: $\vec{v} := \vec{x}_{best} + F\cdot(\vec{x}_{r1} - \vec{x}_{r2}) + F\cdot(\vec{x}_{r3} - \vec{x}_{r4})$; ▷ F is mutation factor in [0,2] and $\vec{x}_{r1}$, $\vec{x}_{r2}$, $\vec{x}_{r3}$, and $\vec{x}_{r4}$ are randomly selected individuals (indices of i, $r1$, $r2$, $r3$, and $r4$ are distinct)
3: Generate a trial vector $\vec{u}$ as follows:
$u^d := \begin{cases} v^d, & \text{if } r <= CR \text{ or } d = I \\ x^d, & \text{if } r > CR \text{ and } d != I \end{cases}$, where CR is a probability constant and I is a random integer within [1,D].
4: **if** $f(\vec{u}) < f(\vec{x}_i)$ **then** $\vec{x}_i := \vec{u}$; **end if**
5: **end for**

Algorithm 5 AMP()

1: $\mathbb{P} \leftarrow 0$; $\mathbb{D} \leftarrow 0$; $t \leftarrow 0$; ▷ *gSize* is an initial population size.
2: Create an initial population C with *gSize* individuals;
3: $\mathbb{P} \leftarrow Cluster(C)$; ▷ Call Algorithm 1.
4: **while** stopping criteria are not satisfied **do**
5: **for** each population $\mathbb{P}[i]$ **do**
6: $\mathbb{P}[i].search()$; ▷ Call Algorithm 3 or Algorithm 4.
7: **if** $\mathbb{P}[i]$ is stagnating **then**
8: $x_{best}^{\mathbb{P}[i]}.cauchy()$; ▷ Call the Cauchy movement by (3).
9: **else**
10: $x_{best}^{\mathbb{P}[i]}.brownian()$; ▷ Call the Brownian movement by (2).
11: **end if**
12: **end for**
13: Hibernate populations when they find new peaks;
14: Degrade individuals if they fall into the attraction area of any peak;
15: Remove excluded populations;
16: **if** $r_{avg}^{conv} < \theta \cdot S$ **then**
17: Wake up hibernating populations;
18: Create a map item and put it to database $\mathbb{D}$;
19: Estimate the number of individuals for the next phase;
20: Create a random population C';
21: $\mathbb{P} \leftarrow \mathbb{P} \cup Cluster(C')$;
22: $t \leftarrow t + 1$; ▷ Increase the phase counter by one.
23: **end if**
24: **end while**

2) AMP With DE: The DE with DE/best/2/bin mutation strategy is used in this paper. In the DE, each donor vector is generated based on the best individual; therefore, all individuals will quickly converge at the location of the best individual as the PSO does. Algorithm 4 shows the procedures of the algorithm for minimization problems. Parameters F and CR are set to 0.5 and 0.6, respectively, which were suggested by [25].

H. Workflow of the AMP Framework

Algorithm 5 presents the workflow of the AMP framework. For each population, either the PSO or DE algorithm is called at step 6, then its best individual undergoes the Brownian or Cauchy movement depending on the status of the population (step 8 or step 10). A population will hibernate if it finds a new peak (step 13). All individuals undergo the degrading process if they fall in the basin of attraction of any explored peak (step 14). Excluded populations will be removed at step 15. The adaptation mechanism is triggered whenever the average radius of nonstagnating populations is less than a small value of $\theta \cdot S$ (step 16). Then, Algorithm 2 is called to estimate the number of individuals for the next phase. A random population (C') is created and clustered into a number of small populations, which are appended to a list ($\mathbb{P}$).

IV. Experimental Studies

To investigate the performance of the AMP framework, a set of experiments are carried out in dynamic and static environments, respectively. For DOPs, ten peer MPMs are selected. They are mQSO [6], SAMO [3], SPSO [30], AMSO [20], CPSO [44], CPSOR [19], FTMPSO [46], DynDE [25], DynPopDE [37], and mNAFSA [45]. The two proposed algorithms are named AMP/PSO and AMP/DE, respectively. Among these algorithms, AMP/PSO, AMP/DE, SAMO, DynPopDE, and AMSO are adaptive algorithms in terms of the number of populations used in the run time. The comparison is conducted based on the MPB problem [8]. For experiments in static environments, ten multimodal problems are chosen for investigating the working mechanisms of the AMP framework.

A. Problem Description

1) Moving Peaks Benchmark: The MPB problem [8] is constructed by a number of peaks, which change in the location, height, and width. For the D-dimensional landscape, the problem is defined as

$$F(\vec{x}, t) = \max_{i=1,\ldots,P} \frac{H_i(t)}{1 + W_i(t) \sum_{j=1}^{D} \left(x_j(t) - X_{ij}(t)\right)^2} \tag{5}$$

where $W_i(t)$ and $H_i(t)$ are the height and width of peak i at time t, respectively, and $X_{ij}(t)$ is the jth element of the location of peak i at time t. The P independently specified peaks are blended together by the max function. The position of each peak is shifted in a random direction by a vector $\vec{v}_i$ of a distance s (s is also called the shift length, which determines the severity of the problem dynamics), and the move of a single peak can be described as

$$\vec{v}_i(t) = \frac{s}{|\vec{r} + \vec{v}_i(t-1)|}((1-\lambda)\vec{r} + \lambda\vec{v}_i(t-1)) \tag{6}$$

where the shift vector $\vec{v}_i(t)$ is a linear combination of a random vector $\vec{r}$ and the previous shift vector $\vec{v}_i(t-1)$ and is normalized to the shift length s. The correlated parameter λ is set to 0, which implies that the peak movements are uncorrelated. A change of a single peak can be described as

$$H_i(t) = H_i(t-1) + \text{height\_severity}_i * \sigma \tag{7a}$$

$$W_i(t) = W_i(t-1) + \text{width\_severity}_i * \sigma \tag{7b}$$

$$\vec{X}_i(t) = \vec{X}_i(t)(t-1) + \vec{v}_i(t) \tag{7c}$$

where σ is a normal distributed random number with mean 0 and variation 1.

In this paper, a new feature, the change in the number of peaks [20], is used to test an algorithm's performance in terms of adaptation. If this feature is enabled, the number of peaks changes using one of the following:

$$\text{Var1} : P = P + \text{sign} \cdot 2 \tag{8a}$$

$$\text{Var2} : P = P + \text{sign} \cdot r(1, 5) \tag{8b}$$

$$\text{Var3} : P = r(10, 100) \tag{8c}$$

where sign $= 1$ if $P <= 10$, sign $= -1$ if $P >= 100$, and the initial value of sign is one; $r(a, b)$ returns a random value in $[a, b]$. The default settings for the MPB are given in Table I.

TABLE I
Default Settings for the MPB, Where u Means That the Problem Changes Every u Objective Evaluations, R Denotes the Range of Allele Values, and I, H, W Denote the Initial Height, Height Range, and Width Range, Respectively, for All Peaks

Parameter	Value	Parameter	Value
number of peaks (P)	10	number of dimensions (D)	5
change frequency (u)	5000	correlation coefficient (λ)	0
height severity	[1,10]	number of peaks change	no
width severity	[0.1,1.0]	R	[0, 100]
peak shape	cone	H	[30, 70]
basic function	no	W	[1, 12]
shift length (s)	1.0	I	50.0

2) Multimodal Functions: In addition to the MPB problem, ten static functions for multimodal optimization [31], [32] are chosen to comprehensively investigate the effect of various components of the AMP framework. Table II gives a description of these functions, where the major properties are as follows.

1) The waves function (F1) is asymmetric and has ten peaks (one global optimum and nine local optima), which are irregularly placed. Some of the peaks are difficult to find as they lie on the border or on flat hills.
2) The Vincent function (F2) has 6^D global optima and no local optima. The global optima have vastly different spacings between them. A part of the optima are very difficult to find as they take a very narrow space in the fitness landscape.
3) The six-hump camel back function (F3) has six optima, which are placed in a smooth landscape, and two of the optima are global optima.
4) The Shubert function (F4) contains $D3^D$ global optima unevenly distributed. These global optima are divided into 3^D groups, with each group having D global optima that are close to each other. F4 also contains many other local optima, which are between the global optima.
5) The modified Shekel function (F5) has eight optima, one of which is the global optimum. Most of the optima are separated from each other by wide flat valleys. The parameters a_{ij} and c_j are given by

$$a_{ij} = \begin{pmatrix} 4 & 4 & 6.3 & 4 & 4 \\ 1 & 1 & 8.5 & 1 & 1 \\ 6 & 6 & 9.1 & 6 & 6 \\ 3.5 & 7.5 & 4 & 9 & 4 \\ 5 & 5 & 3 & 3 & 9 \\ 9.1 & 8.2 & 2 & 3 & 9 \\ 1.5 & 9.3 & 7.4 & 3 & 9 \\ 7.8 & 2.2 & 5.3 & 9 & 3 \end{pmatrix}$$

$$c_j = \left(0.10.20.40.150.60.20.060.18\right).$$

The location of the global optimum is given by vector row $a_{7j}, j \in [1, D]$, and the other seven local optima are determined by other vector rows of the matrix $||a||$.

6) The IBA function (F6) has three global optima and one local optimum. In the function, $k = -0.95$ and $\chi = -1.26$ is used.

TABLE II
DESCRIPTION OF TEN MULTIMODAL FUNCTIONS, WHERE D IS THE NUMBER OF DIMENSIONS

Name	D	Function	# of global/local Opt.
Waves	2	$F1(\vec{x}) = (0.3x_1)^3 + 3.5x_1x_2^2 - 4.7cos(3x_1 - x_2^2(2+x_1))sin(2.5\pi x_1)$, $-0.9 \le x_1 \le 1.2, -1.2 \le x_2 \le 1.2$	1/9
Vincent	D	$F2(\vec{x}) = (\sum_{i=1}^{D} sin(10logx_i))/D$, $0.25 \le x_i \le 10$	6^D/0
Six-hump Camel Back	2	$F3(\vec{x}) = (4 - 2.1x_1^2 + x_1^4/3)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2)$, $-1.9 \le x_1 \le 1.9, -1.1 \le x_2 \le 1.1$	2/4
Shubert	D	$F4(\vec{x}) = \Pi_{j=1}^{D}(\sum_{i=1}^{5} icos((i+1)x_j + i))$, $-10 \le x_i \le 10$	$D3^D$/many
Modified Shekel	D	$F5(\vec{x}) = \sum_{i=1}^{8}(\sum_{j=1}^{D}(x_j - a_{ij})^2 + c_j)^{-1}$, $0 \le x_i \le 11$	1/7
IBA	2	$F6(\vec{x}) = (x_1^2 + x_2^2)/(1 + x_1^2 + x_2^2) + k(14(x_1^2 + x_2^2) + (x_1^2 + x_2^2)^2\chi^2 - 2\sqrt{14}(x_1^3 - 3x_1x_2^2)\chi)/(14(1 + x_1^2 + x_2^2)^2)$ $-4 \le x_i \le 4$	3/1
Himmenblau	2	$F7(\vec{x}) = (x_1^2 + x_2 - 11)^2 + (x_2^2 + x_1 - 7)^2 + 0.1((x_1 - 3)^2 + (x_2 - 2)^2)$, $-6 \le x_i \le 6$	1/3
Five hills	2	$F8(\vec{x}) = sin(2.2\pi x_1 + 0.5\pi)(2 - \lvert x_2\rvert)/2(3 - \lvert x_1\rvert)/2 + sin(0.5\pi x_2 + 0.5\pi)(2 - \lvert x_2\rvert)/2(2 - \lvert x_1\rvert)/2$ $-2.5 \le x_1 \le 3, -2 \le x_2 \le 2$	1/4
Center peak	2	$F9(\vec{x}) = 3sin(0.5\pi x_1 + 0.5\pi)(2 - \sqrt{x_1^2 + x_2^2})/4$, $-2 \le x_i \le 2$	1/4
BraninRCOS	2	$F10(\vec{x}) = (x_2 - 5.1 * x_1^2/(4\pi^2) + 5 * x_1/\pi - 6)^2 + 10(1 - 1/(8\pi))cosx_1 + 10$, $-5 \le x_1 \le 10, 0 \le x_2 \le 15$	3/0

TABLE III
THRESHOLD VALUES OF ϵ_s AND ϵ_o FOR IDENTIFYING OPTIMA, WHERE GO IS THE OBJECTIVE VALUE OF THE GLOBAL OPTIMA

	MPB	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
ϵ_s	-	0.15	0.2	0.1	0.5	0.2	0.08	0.5	0.2	0.2	1.0
ϵ_o	0.1	1E-3	1E-4	1E-4	1E-3	1E-3	1E-6	1E-4	1E-4	1E-5	1E-5
GO(D=2)	-	7.307	1	-1.0316	186.731	16.832	-8.016E-3	2	2.5	1.5	0.397887

7) The Himmenblau function (F7) has one global optimum and three local optima with different objective values.
8) The five hills function (F8) and the center peak function (F9) each have one global optimum and four local optima. F8 has five very close hills with lines of valleys between them. The four local optima in F9 are on the edge of the intervals, and the global optimum is in the middle.
9) The Branin RCOS function (F10) has three global optimum, which are distributed within an irregular and asymmetric landscape. In addition, the function has no local optima.

B. Experimental Setup

1) Performance Evaluation: To evaluate an algorithm's performance in tracking the global optimum, the offline error (E_O) [10] and the best-before-change error (E_{BBC}) are used. The offline error used in this paper is the average of the best error found every two objective evaluations, and the best-before-change error is the average of the best error achieved at the fitness evaluation just before a change occurs.

In addition, to evaluate an algorithm's performance in tracking multioptima, the ratio of peaks that are traced (PR) and the success rate (SR) of tracking all peaks are used. A peak is assumed to be traced if the difference of objective values between any individual and the peak is less than ϵ_o, and the Euclidian distance between the individual and the peak is less than ϵ_s. (See Table III for the values of ϵ_o and ϵ_s for all problems tested in this paper.) For the value of ϵ_s for the MPB, it is set to $\min(\min_{i \ne j \le P} d(X_i(t), X_j(t))/2, 0.1)$ at time t.

A two-tailed t-test with 58 degrees of freedom at a 0.05 level of significance was conducted for each pair of algorithms on E_O and E_{BBC}. The t-test results are given with the letters "w," "l," or "t," which denote that the performance of an algorithm is significantly better than, significantly worse than, or statistically equivalent to its peer algorithms, respectively.

2) Algorithm Configurations: For the AMP framework, there are three parameters to be investigated (see Section IV-C later). They are the initial population size (gSize), the convergence threshold (θ), and the probabilistic range M, whose default values are set to 100, $0.005 \times S$, and 3, respectively. For parameters of all the peer algorithms, default values suggested in their proposals are used if the algorithms show their best results. For example, the equations of setting the exclusion radius for each population suggested by Blackwell *et al*. [3], [6] work well in our test. Therefore, we also use the default setting regarding this parameter for mQSO [6], SAMO [3], SPSO [30], DynDE [25], mNAFSA [45], and DynPopDE [37] as their authors used. However, for mNAFSA [45] try_number $= 2$ and $N = 10$ are used instead of try_number $= 4$ and $N = 2$ suggested in [45]. The stopping criterion is 200 changes for the MPB problem and 2.0E+5 function evaluations, or finding all peaks for multimodal problems in static environments. All the results reported in the paper are averaged over 30 independent runs of an algorithm on each problem.

3) Open Frameworks for Evolutionary Computation: The open frameworks for evolutionary computation (OFEC) is a template library written in C++. It supports any population-based EC methods running in parallel. The source code of the AMP framework is available in OFEC. The OFEC-v0.4.0 has been released on github at https://github.com/Changhe160/OFEC.

C. Investigation of Parameters of the AMP Framework

In this section, several sensitivity analyses are carried out to investigate the key parameters of the AMP framework. In this experiment the chosen search mechanism is the PSO algorithm.

1) Sensitivity Analysis of Parameter gSize: To test the effect of varying the initial population size, gSize was chosen from 10 to 500. Fig. 4 presents the results of E_O, E_{BBC}, PR, and SR of AMP/PSO with different values of gSize on the MPB with different numbers of peaks.

Fig. 4 shows that varying the value of gSize in all the cases does not affect the results too much. This indicates that AMP/PSO has a very stable performance regardless of the initial population size. We would attribute the stable performance of AMP/PSO to its adaptation mechanism. The continuous adjustment to historical data [see (1)] would enable AMP/PSO to adjust the number of populations to an appropriate level

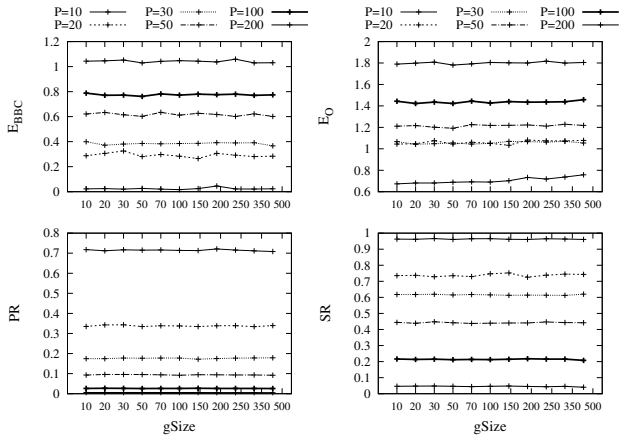

Fig. 4. Results of AMP/PSO with different initial sizes (gSize) on the MPB with different numbers of peaks.

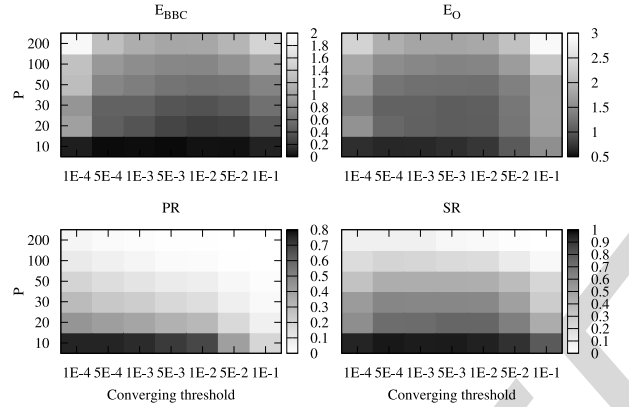

Fig. 5. Results of AMP/PSO with different converging thresholds on the MPB with different numbers of peaks.

(see evidence in Figs. 6, 10, and 11 later). Therefore, different choices of gSize do not affect the performance of AMP/PSO, and $g\text{Size} = 100$ is used for both AMP algorithms in this paper later on.

2) Sensitivity Analysis of Parameter θ: For the convergence threshold (θ), the smaller it is, the more time will be spent on exploiting local optima. By contrast, the larger the value of θ, the more time will be spent on exploring new optima. To investigate the effect of this parameter on the performance of AMP/PSO, we conduct an experiment with different values of θ. Fig. 5 presents the results of AMP/PSO with different values of θ on the MPB with different numbers of peaks, where the darker the shade is, the better the results are.

Fig. 5 shows that varying the value of θ does affect the performance of AMP/PSO. For E_O, E_{BBC}, and SR, in most cases the results get better as θ increases from $1E$-4 to $5E$-3 but then get worse as θ further increases. However, for PR, the smallest value of θ helps AMP/PSO obtain the best performance, and the results of PR get worse as θ increases. This is because the measurement PR focuses more on exploitation of local optima than on exploration of the global optima. AMP/PSO will spend the largest amount of time on exploiting local optima with the smallest value of θ and hence the largest number of peaks is tracked. To take a trade-off between exploring new optima and exploiting local optima, θ was set to $5E$-3 for the AMP framework in all the other experiments in this paper.

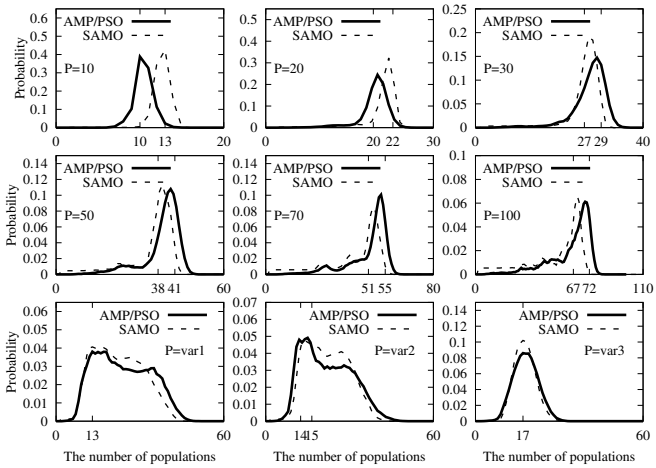

Fig. 6. Distribution of changes of the number of populations of AMP/PSO and SAMO over 1000 changes.

TABLE IV
EFFECT OF VARYING THE PROBABILISTIC RANGE M ON THE MPB WITH DIFFERENT NUMBERS OF PEAKS, WHERE WAR IS A WRONG ACTION RATE AND NR IS THE NUMBER OF MAP ITEMS CREATED IN THE DATABASE $\mathbb{D}$

P	WAR/NR				
	M=1	M=2	M=3	M=4	M=5
10	0.295/3720	0.188/3864	0.126/4027	0.097/4189	0.069/4239
20	0.320/2689	0.214/2904	0.128/2977	0.114/2985	0.091/2995
30	0.327/2000	0.224/2176	0.153/2310	0.114/2371	0.103/2292

TABLE V
PERFORMANCE COMPARISON OF AMP/PSO WITH DIFFERENT VALUES OF M ON MULTIMODAL FUNCTIONS, WHERE RE IS THE RATIO OF EVAL TO THE LARGEST EVAL OF EACH PROBLEM, NR IS THE NUMBER OF MAP ITEMS CREATED IN THE DATABASE $\mathbb{D}$

Error	F(Opt.)	M=1	M=2	M=3	M=4	M=5	F(Opt.)	M=1	M=2	M=3	M=4	M=5
SR		**1**	**1**	**1**	**1**	**1**		0.967	**1**	**1**	**1**	**1**
RE	F1(10)	0.981	1	1	**0.962**	0.98	F2(36)	0.996	1	0.936	0.935	**0.884**
NR		33	47	26	**6**	23		194	132	66	99	**44**
SR		**1**	**1**	**1**	**1**	**1**		**1**	**1**	**1**	**1**	**1**
RE	F3(6)	0.945	0.941	**0.938**	0.977	1	F4(18)	0.945	0.933	**0.835**	1	0.835
NR		27	22	21	18	**10**		**51**	67	56	88	60
SR		**1**	**1**	**1**	**1**	**1**		**1**	**1**	**1**	**1**	**1**
RE	F5(8)	0.939	0.879	**0.874**	0.931	1	F6(4)	1	0.961	**0.92**	0.985	0.95
NR		61	38	**26**	46	83		40	**25**	34	29	29
SR		**1**	**1**	**1**	**1**	**1**		**1**	**1**	**1**	**1**	**1**
RE	F7(4)	1	0.972	0.965	**0.956**	0.966	F8(5)	1	**0.96**	0.976	0.967	0.965
NR		**3**	**3**	**3**	**3**	**3**		39	23	**11**	18	18
SR		**1**	**1**	**1**	**1**	**1**		**1**	**1**	**1**	**1**	**1**
RE	F9(5)	0.997	1	0.998	**0.978**	0.981	F10(3)	1	0.999	0.952	**0.932**	0.936
NR		13	11	14	**3**	8		4	**3**	4	4	6

3) Sensitivity Analysis of Parameter M: To show the impact of using different values of the probabilistic range (M), an experiment was carried out to calculate the average wrong action rate for AMP/PSO with M in [1, 5] on the MPB problem with $u = 10\,000$. According to the results in Fig. 6 (to be introduced later), for problems with $P \leq 30$, we assume that the optimal number of populations is equal to the number of peaks. Hence, a wrong action is an action to increase/decrease the total number of individuals when the number of peaks is less/greater than the current number of populations. Table IV presents the wrong action rate (WAR) and the number of map

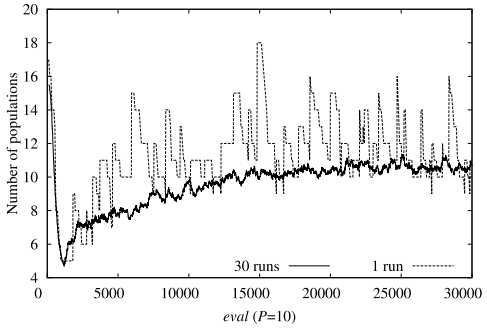

Fig. 7. Number of populations against time on the ten-peak MPB problem with six changes for a single run and multiple runs.

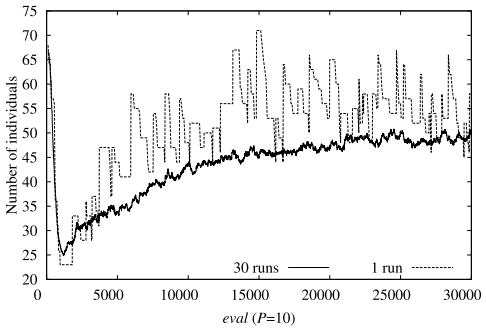

Fig. 8. Number of individuals against time on the ten-peak MPB problem with six changes for a single run and multiple runs.

items created in the database $\mathbb{D}$. The results show that using the probabilistic scheme $M > 1$ does help AMP/PSO to greatly decrease the probability of taking a wrong action.

To further investigate the impact of using different values of M, an experiment was carried out on the ten static problems. Table V presents the results, where RE is the ratio of the number of evaluations (eval) to the largest eval of each test group. Table V shows that AMP/PSO fails to find all the global optima of F2 with $M = 1$. For most problems AMP/PSO with $M = 1$ generates the largest number of map items in the database $\mathbb{D}$, and correspondingly, it also spends a relatively large number of evaluations. Although less map items are generated with $M > 1$ than that with $M = 1$, they enable AMP/PSO to spend less time to find all the optimal peaks than that with $M = 1$. That is, the probabilistic prediction scheme makes the learning more efficient than that without it. In this paper, $M = 3$ is used for all the experiments, as AMP/PSO with $M = 3$ shows the best performance on most problems.

D. Investigation of Components of the AMP Framework

In this section, the effect of each component of the AMP framework is investigated based on AMP/PSO.

1) Effect of the Adaptation Scheme: Fig. 6 shows the distribution of the number of populations of AMP/PSO and SAMO over 1000 changes. For AMP/PSO, we use all the map items stored in the database as sample data. For SAMO, we take one sample when the number of populations changes. Both algorithms are able to maintain a good correlation between the number of populations and the number of peaks. In problems in which the number of peaks remain unchanged, each distribution curve has a very narrow shape. However, the performance of AMP/PSO is better than that of SAMO at least on problems with a small number of peaks (e.g., $P = 10$ and 20), where the numbers of populations of AMP/PSO at the curve peaks are equal to the actual numbers of peaks. In contrast, the numbers of population obtained by SAMO are larger than the actual numbers of peaks. For problems with a large number of peaks, the numbers of populations at the curve peaks for both algorithms are smaller than the actual numbers of peaks. However, the numbers of populations for AMP/PSO are closer to the actual numbers of peaks.

In problems with a varying number of peaks by a certain pattern (Var1 and Var2), for both algorithms the distribution curves now have a large band with multiple spikes, corresponding to the variation in the number of peaks of these problems. However, both algorithms do not show such a behavior in Var3 where the number of peaks changes without a pattern (further comparison of detailed changes in the number of populations can be seen later in Figs. 10 and 11).

2) Effect of the Clustering Method and the Population Exclusion Scheme: To show the effect of the clustering method and the population exclusion scheme, an experiment was carried out on the ten-peak MPB problem with six environmental changes. Fig. 7 presents the change in the number of populations against time for a single run and for 30 runs (on average). Fig. 8 presents the change in the number of individuals for a single run and for 30 runs (on average). From the curves of the single run, initially 17 populations are obtained after clustering 100 random individuals. Note that the total number of individuals as shown in Fig. 8 with the 17 populations is 68, not the initial value of 100. This is because a part of the populations obtained by the clustering method do not satisfy the minimum population size. As mentioned in Section III-A, such populations will be ignored. As the run goes on, the number of populations gradually decreases due to the removal of excluded populations, and only five populations with a total of 23 individuals survive by the time point of the first population adjustment.

The clustering operation is performed whenever a new population is created due to the trigger of the adaptation scheme. The population exclusion scheme is performed whenever overcrowding populations are detected. Clustering a certain number of new random individuals helps AMP/PSO find unexplored peaks. The removal of excluded populations can greatly save computing resources. More important, it enables the adaptation scheme to obtain an appropriate number of populations for a particular problem. The curve of the 30 runs in Fig. 7 shows that the average number of populations gradually converges at a certain level, which is close to the number of peaks of the ten-peak MPB problem.

3) Effect of the Peak Hiding Scheme: The peak hiding scheme encourages individuals to explore undiscovered peaks and hence improves the search efficiency. Table VI presents the comparison of AMP/PSO with the peak hiding scheme and without the peak hiding scheme (AMP/PSO$^{-\text{ph}}$) on the Vincent (F2) and Shubert (F4) functions in the 2-D space. F2 and F4 contain 36 and 18 global optima, respectively.

TABLE VI
COMPARISON OF SR AND PR OF AMP/PSO WITH AND WITHOUT THE PEAK HIDING SCHEME ON F2 AND F4, WHERE EVAL$*$ IS THE NUMBER OF EVALUATIONS USED BY THE PEAK HIDING SCHEME AND EVAL IS THE TOTAL NUMBER OF EVALUATIONS

Function	Algorithm	SR	PR	$eval$	$eval*$
Vincent(F2,D=2)	AMP/PSO^{-ph}	0.1	0.95	191937	0
	AMP/PSO	1	1	140139	102871
Shubert(F4,D=2)	AMP/PSO^{-ph}	0.9	0.99	82437	0
	AMP/PSO	1	1	39144	11138

TABLE VII
COMPARISON BETWEEN AMP/PSO WITH AND WITHOUT THE BROWNIAN MOVEMENT ON THE MPB PROBLEM WITH DIFFERENT NUMBERS OF PEAKS, WHERE N^e IS THE AVERAGE NUMBER OF POPULATIONS AT THE END OF EACH EVOLVING PHASE

Algorithm		$P=10$	$P=20$	$P=30$	$P=50$
AMP/PSO^{-bm}	E_{BBC}	1.69±0.32	3.11±0.33	1.65±0.24	1.86±0.17
	E_O	3.26±0.30	3.98±0.31	2.52±0.21	2.57±0.17
	PR	0.16	0.06	0.04	0.025
	N^e	15.48	22.58	25.92	33.23
AMP/PSO	E_{BBC}	0.016±0.01	0.28±0.1	0.38±0.04	0.61±0.05
	E_O	0.69±0.03	1.1±0.09	1.1±0.05	1.2±0.05
	PR	0.71	0.34	0.18	0.092
	N^e	10.56	17.07	22.30	29.32

TABLE VIII
COMPARISON OF PR AND SR BETWEEN AMP/PSO WITH AND WITHOUT THE CAUCHY MOVEMENT ON F2 AND F4 IN 3-D AND 4-D SPACE

PR/SR	$F2(3D)$	$F4(3D)$	PR/SR	$F2(3D)$	$F4(3D)$
AMP/PSO^{-cm}	0.762/0	0.70/0	AMP/PSO	0.78/0	0.72/0

TABLE IX
COMPARISON BETWEEN AMP/PSO WITH AND WITHOUT THE HIBERNATION SCHEME ON THE MULTIMODAL PROBLEMS, WHERE EVAL IS THE NUMBER OF EVALUATIONS

Function	AMP/PSO^{-hw}			AMP/PSO		
	PR	SR	$eval$	PR	SR	$eval$
F1	1	1	1.7E+4	1	1	1.5E+4
F2(2D)	1	1	2.0E+5	1	1	1.44E+5
F3	1	1	2.6E+4	1	1	2.17E+4
F4(2D)	1	1	5.0E+4	1	1	4.0E+4
F5(2D)	1	1	7.2E+4	1	1	5.96E+4
F6	1	1	1.03E+5	1	1	9.17E+4
F7	1	1	1.3E+4	1	1	1.1E+4
F8	1	1	2.12E+4	1	1	1.69E+4
F9	1	1	6.78E+3	1	1	6.2E+3
F10	1	1	6.35E+3	1	1	5.07E+3

TABLE X
PR AND SR OF AMP/PSO ON FUNCTIONS F2 AND F4 IN 3-D AND 4-D SPACE, WHERE GOPT IS THE NUMBER OF GLOBAL OPTIMA, EVAL IS THE NUMBER OF FUNCTION EVALUATIONS, AND THE MAXIMUM NUMBER OF EVALUATIONS FOR EACH PROBLEM IN 3-D AND 4-D ARE 5.0E+06 AND 1.0E+07, RESPECTIVELY

F(D,$GOpt$)	PR	SR	$eval$	F(D,$GOpt$)	PR	SR	$eval$
F2(3,216)	1	1	1.25E+06	F2(4,1296)	0.95	0	1.0E+07
F4(3,81)	1	1	3.89E+06	F4(4,324)	0.828	0	1.0E+07

Table VI shows that AMP/PSO finds all peaks on the two functions for all runs. However, the results of SR and PR of AMP/PSO get worse when the peak hiding scheme is disabled. The only disadvantage of the peak hiding scheme is that extra evaluations are needed. However, the total function evaluations are still less than that of AMP/PSO$^{-\text{ph}}$. Moreover, the peak hiding scheme greatly improves the performance of AMP/PSO. The peak hiding scheme is rarely triggered for the MPB problem, as there is not enough time for populations to converge before changes occur. Therefore, the peak hiding scheme has little effect on the MPB problem.

4) Effect of the Brownian and Cauchy Movements: Table VII shows the comparison between AMP/PSO with and without (AMP/PSO$^{-\text{bm}}$) the Brownian movement on the MPB problem with different numbers of peaks. From the table, it can be seen that the performance of AMP/PSO$^{-\text{bm}}$ is much worse than that of AMP/PSO in all cases. The explanation can be obtained from the comparison of the results of the number of survived populations (N^e) between the two algorithms. The number of survived populations of AMP/PSO$^{-\text{bm}}$ is larger than that of AMP/PSO in all cases. As discussed above in Section III-F, in AMP/PSO$^{-\text{bm}}$ more than one population may converge on the slope of the same peak, thus the number of converging populations will be larger than it is supposed to be. This will cause two issues. First, the error of E_{BBC} will increase as the highest peak may not be sufficiently exploited if there is no population converging on its summit. Second, the peak ratio (PR) will decrease as tracking peaks will become inefficient if a larger number of populations than necessary is used, e.g., the value of N^e is much larger than the total number of peaks on the ten-peak MPB problem.

To investigate the effect of the Cauchy movement, an experiment was carried out on AMP/PSO with and without (AMP/PSO$^{-\text{cm}}$) the Cauchy movement on the Vincent (F2) and Shubert (F4) functions in the 3-D space. Table VIII presents the comparison results of PR and SR. As discussed above in Section III-F, the Cauchy movement is helpful for stagnating populations to jump out of their trapped areas and hence more peaks would be explored. This can be observed in Table VIII, where AMP/PSO obtains more peaks than AMP/PSO$^{-\text{cm}}$.

5) Effect of the Hibernation Scheme: The motivation of the hibernation scheme is to save function evaluations. To see its effect, the total number of function evaluations of AMP/PSO with and without (AMP/PSO$^{-\text{hw}}$) this scheme is compared on the ten multimodal problems. Table IX presents the comparison results of the number of evaluations (eval), PR, and SR. From the results, it can be seen that both algorithms achieve the same results on PR and SR. However, AMP/PSO spends less function evaluations than AMP/PSO$^{-\text{hw}}$ on all problems.

6) Locating Many Peaks in Static Environments: In order to investigate the capability of locating many peaks, AMP/PSO is applied to the Vincent (F2) and Shubert (F4) functions in the 3-D and 4-D spaces with the maximum number of function evaluations of 5.0E+06 and 1.0E+07, respectively. Table X shows the results of PR and SR. For both problems in 3-D space, AMP/PSO successfully finds all peaks under the given maximum number of evaluations. For the 4-D-Vincent function with 1296 global optima, although AMP/PSO fails to find all peaks under the given maximum number of evaluations, it achieves a PR score of 0.95. AMP/PSO also achieves a PR score of 0.82 on the 4-D Shubert function with 324 global optima.

TABLE XI
COMPARISON OF ERRORS OF E_O AND E_{BBC} ON THE MPB PROBLEM WITH DIFFERENT NUMBERS OF PEAKS

P	error	AMP/PSO	AMP/DE	SAMO	DynPopDE	SPSO	mQSO	CPSOR	CPSO	FTMPSO	DynDE	AMSO	mNAFSA
10	E_O	**0.69**±0.03	0.9±0.1	2.4±0.3	4.8±1	4.4±0.5	1.9±0.3	5±0.3	5.2±0.3	2.6±0.2	2.1±0.3	2.5±0.5	3.4±0.5
	w,t,l	11,0,0	10,0,1	6,1,4	0,2,9	3,0,8	9,0,2	1,1,9	0,1,10	5,1,5	8,0,3	5,2,4	4,0,7
	E_{BBC}	**0.016**±0.01	0.069±0.1	1.2±0.3	3.5±1	3.1±0.5	0.8±0.3	0.96±0.3	1±0.3	1.2±0.3	1.2±0.3	1.1±0.6	1.7±0.5
	w,t,l	11,0,0	10,0,1	3,3,5	0,1,10	0,1,10	9,0,2	6,2,3	4,4,3	3,4,4	3,4,4	3,5,3	2,0,9
20	E_O	**1.1**±0.09	1.4±0.1	2.3±0.08	5.1±1	7.2±0.9	3.3±0.4	4±0.2	4.9±0.2	3.2±0.2	2.6±0.3	2.8±0.8	4.8±0.6
	w,t,l	11,0,0	10,0,1	9,0,2	1,2,8	0,0,11	5,1,5	4,0,7	1,2,8	5,1,5	7,1,3	7,1,3	1,2,8
	E_{BBC}	**0.28**±0.1	0.52±0.2	1.3±0.1	3.7±1	6.4±1	2.5±0.4	1.2±0.2	2±0.2	2.3±0.2	1.9±0.3	1.5±0.9	3.6±0.6
	w,t,l	11,0,0	10,0,1	7,1,3	1,1,9	0,0,11	3,0,8	9,0,2	5,1,5	4,0,7	5,1,5	7,1,3	1,1,9
100	E_O	**1.4**±0.05	1.7±0.08	2.1±0.06	4.3±0.8	5.4±1	3.3±0.4	2.4±0.07	3.2±0.1	3.2±0.1	3.5±0.6	2.2±0.4	4±0.2
	w,t,l	11,0,0	10,0,1	8,1,2	1,1,9	0,0,11	3,2,6	7,0,4	5,1,5	4,2,5	3,1,7	8,1,2	1,1,9
	E_{BBC}	**0.77**±0.04	0.9±0.07	1.4±0.06	2.7±0.6	4.5±1	2.6±0.4	1±0.06	1.4±0.08	2.5±0.1	2.9±0.6	1.1±0.5	2.7±0.2
	w,t,l	11,0,0	10,0,1	6,0,5	1,4,6	0,0,11	1,4,6	8,1,2	7,0,4	3,2,6	1,3,7	8,1,2	1,3,7
200	E_O	**1.8**±0.04	2±0.04	2.5±0.05	4.6±0.7	6.2±0.9	4.3±0.4	2.5±0.08	3.1±0.07	3.6±0.1	4.4±0.4	2.5±0.3	4.2±0.2
	w,t,l	11,0,0	10,0,1	8,1,2	1,1,9	0,0,11	2,2,7	7,1,3	6,0,5	5,0,6	1,2,8	7,2,2	3,1,7
	E_{BBC}	**1**±0.05	1.2±0.04	1.7±0.04	2.8±0.5	5.4±0.9	3.5±0.4	1.2±0.07	1.4±0.06	2.8±0.1	3.8±0.4	1.3±0.3	2.8±0.2
	w,t,l	11,0,0	9,1,1	6,0,5	3,2,6	0,0,11	2,0,9	9,1,1	7,1,3	3,2,6	1,0,10	7,1,3	3,2,6
w-l		88	71	25	-58	-81	-11	20	-8	-12	-18	30	-46

TABLE XII
COMPARISON OF ERRORS OF E_O AND E_{BBC} ON THE MPB PROBLEM WITH A VARYING NUMBER OF PEAKS

P	error	AMP/PSO	AMP/DE	SAMO	DynPopDE	SPSO	mQSO	CPSOR	CPSO	FTMPSO	DynDE	AMSO	mNAFSA
Var1	E_O	**1.8**±0.1	2±0.07	2.8±0.2	6.3±0.9	6±0.9	3.8±0.4	3.7±0.2	5.1±0.2	4.1±0.2	3.4±0.3	3±0.3	6.6±0.3
	w,t,l	11,0,0	10,0,1	9,0,2	0,2,9	1,1,9	5,0,6	6,0,5	3,0,8	4,0,7	7,0,4	8,0,3	0,1,10
	E_{BBC}	0.99±0.1	**0.99**±0.08	1.8±0.2	4.3±0.6	4.7±0.9	2.9±0.4	1.5±0.2	2.3±0.2	3.1±0.3	2.7±0.3	1.5±0.4	4.8±0.3
	w,t,l	10,1,0	10,1,0	7,0,4	2,0,9	0,1,10	4,0,7	8,1,2	6,0,5	3,0,8	5,0,6	8,1,2	0,1,10
Var2	E_O	**1.4**±0.08	1.6±0.09	2.2±0.1	5.7±1	6.2±0.6	3.3±0.5	3.1±0.1	3.9±0.2	3.1±0.2	3.1±0.3	3.2±2	5.6±0.4
	w,t,l	11,0,0	10,0,1	9,0,2	1,1,9	0,0,11	4,2,5	5,3,3	3,0,8	5,3,3	4,4,3	4,4,3	1,1,9
	E_{BBC}	**0.74**±0.08	0.86±0.1	1.4±0.1	4±1	5.2±0.7	2.6±0.5	1.2±0.08	1.5±0.2	2.5±0.2	2.4±0.3	2±2	3.8±0.4
	w,t,l	11,0,0	10,0,1	6,2,3	1,1,9	0,0,11	3,3,5	9,0,2	6,2,3	3,3,5	3,3,5	3,5,3	1,1,9
Var3	E_O	**1.9**±0.2	2.2±0.2	2.9±0.1	10±1	4.5±0.7	3.3±0.2	4.1±0.2	4.6±0.2	4±0.1	3.3±0.2	3.3±0.5	6.9±0.6
	w,t,l	11,0,0	10,0,1	9,0,2	0,0,11	2,1,8	6,2,3	4,0,7	2,1,8	5,0,6	6,2,3	6,2,3	1,0,10
	E_{BBC}	**1.2**±0.2	1.4±0.2	2±0.1	6.6±0.8	3.5±0.7	2.5±0.2	1.8±0.1	1.9±0.2	3.7±0.3	2.6±0.2	1.9±0.4	4.9±0.6
	w,t,l	11,0,0	10,0,1	6,2,3	0,0,11	2,1,8	4,1,6	9,0,2	6,2,3	2,1,8	4,1,6	6,2,3	1,0,10
w-l		65	55	30	-54	-52	-6	20	-9	-15	2	18	-54

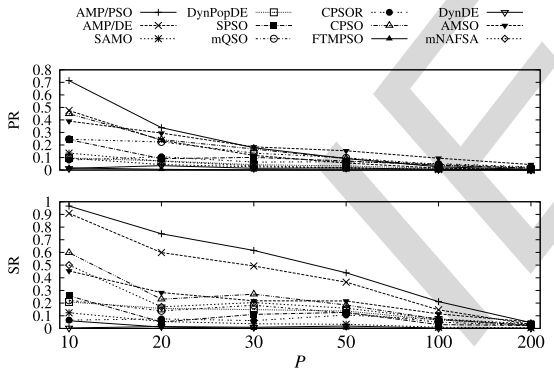

Fig. 9. Comparison of the PR and SR on the MPB with different numbers of peaks.

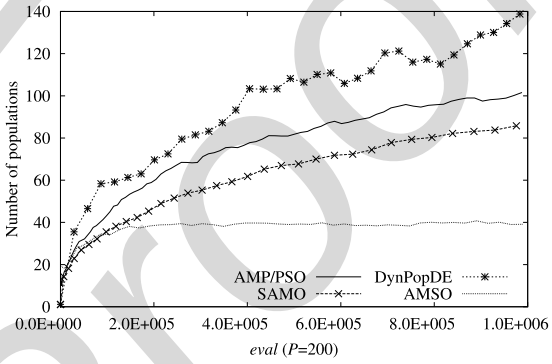

Fig. 10. Changes in the number of populations against time for four adaptive algorithms on the MPB with different numbers of peaks.

E. Comparison With Peer Algorithms on the MPB Problem

1) Effect of Varying the Number of Peaks: Table XI presents the offline errors and the best-before-change errors for all algorithms on the MPB with different numbers of peaks. Fig. 9 presents the comparison of the results of SR and PR. Fig. 10 plots the changes in the number of populations against time for AMP/PSO, SAMO, AMSO, and DynPopDE.

From Table XI and Fig. 9, the results of AMP/PSO and AMP/DE are significantly better than those of the other algorithms in most cases due to the proposed adaptation mechanism. In Table XI, the performance of the four adaptive algorithms (AMP/PSO, AMP/DE, SAMO, and AMSO) is better than that of the nonadaptive algorithms. Due to a large number of populations generated with DynPopDE (Fig. 10), the algorithm achieves very poor performance compared with the other algorithms. Given the manually configured number of individuals, CPSOR also achieves a good performance. Regarding the PR, AMSO performs better than AMP/PSO and AMP/DE in the cases with many peaks. This is because many populations of the AMP framework have not converged yet when changes occur. The number of peaks traced will be increased if we give more time for the AMP framework to evolve before changes occur. (See Fig. 12 in Section IV-E3 later.)

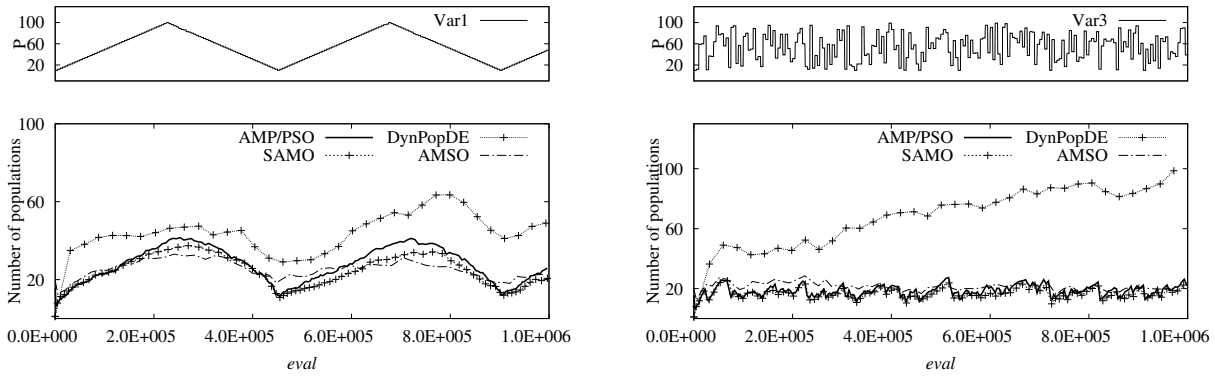

Fig. 11. Changes in number of populations against time for four adaptive algorithms on the MPB, with a varying number of peaks.

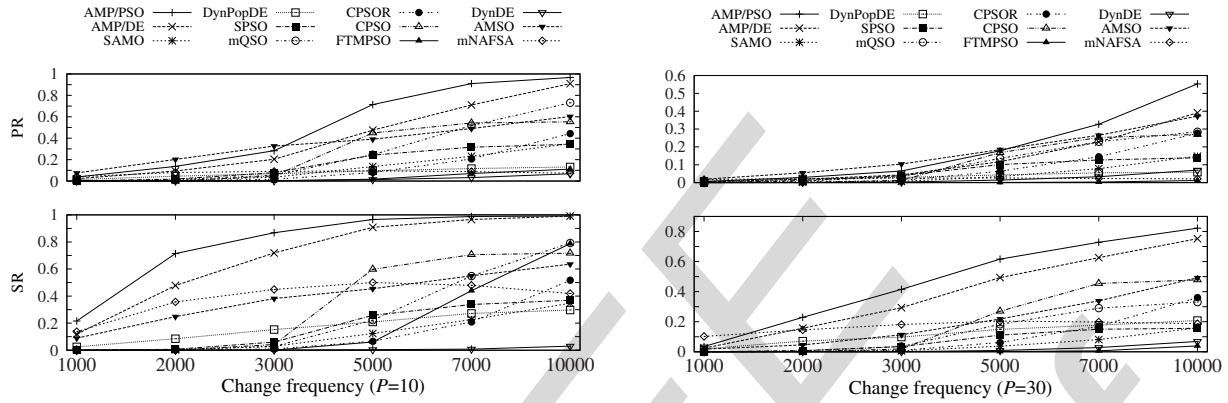

Fig. 12. Comparison of the PR and SR on the MPB with different change frequencies.

TABLE XIII
COMPARISON OF ERRORS OF E_O AND E_{BBC} ON THE 200-PEAK MPB PROBLEM WITH DIFFERENT CHANGE FREQUENCIES

u	error	AMP/PSO	AMP/DE	SAMO	DynPopDE	SPSO	mQSO	CPSOR	CPSO	FTMPSO	DynDE	AMSO	mNAFSA
1000	E_O	**3.6**±0.1	4.1±0.1	4.3±0.1	10±1	7.4±1	6.7±0.9	7.2±0.2	9.7±0.3	8±0.7	5.8±0.6	5.3±0.2	7.3±0.6
	w,t,l	11,0,0	10,0,1	9,0,2	0,1,10	3,2,6	6,0,5	3,2,6	0,1,10	2,0,9	7,0,4	8,0,3	3,2,6
	E_{BBC}	**2.3**±0.1	2.7±0.09	2.9±0.08	7.3±2	6.6±1	5.8±0.9	3.4±0.1	3.9±0.09	7±1	4.7±0.6	2.7±0.1	4.7±0.5
	w,t,l	11,0,0	9,1,1	8,0,3	0,2,9	0,2,9	3,0,8	7,0,4	6,0,5	0,2,9	4,1,6	9,1,1	4,1,6
3000	E_O	**2.2**±0.06	2.5±0.08	2.9±0.07	6.3±1	6.3±0.8	4.4±0.4	3.4±0.09	4.2±0.09	4.4±0.2	4.7±0.4	3±0.3	4.8±0.3
	w,t,l	11,0,0	10,0,1	9,0,2	0,1,10	0,1,10	4,1,6	7,0,4	6,0,5	4,1,6	2,1,8	8,0,3	2,1,8
	E_{BBC}	**1.4**±0.06	1.5±0.07	2±0.06	4±1	5.5±0.9	3.6±0.4	1.6±0.07	2±0.07	3.5±0.1	4±0.4	1.7±0.3	3.1±0.2
	w,t,l	11,0,0	10,0,1	6,1,4	1,2,8	0,0,11	2,2,7	8,1,2	6,1,4	3,1,7	1,1,9	8,1,2	5,0,6
5000	E_O	**1.8**±0.04	2±0.04	2.5±0.05	4.6±0.7	6.2±0.9	4.3±0.4	2.5±0.08	3.1±0.07	3.6±0.1	4.4±0.4	2.5±0.3	4.2±0.2
	w,t,l	11,0,0	10,0,1	8,1,2	1,1,9	0,0,11	2,2,7	7,1,3	6,0,5	5,0,6	1,2,8	7,2,2	3,1,7
	E_{BBC}	**1**±0.05	1.2±0.04	1.7±0.04	2.8±0.5	5.4±0.9	3.5±0.4	1.2±0.07	1.4±0.06	2.8±0.1	3.8±0.4	1.3±0.3	2.8±0.2
	w,t,l	11,0,0	9,1,1	6,0,5	3,2,6	0,0,11	2,0,9	9,1,1	7,1,3	3,2,6	1,0,10	7,1,3	3,2,6
10000	E_O	**1.3**±0.03	1.5±0.04	2±0.03	3.7±0.3	6.2±1	3.7±0.4	2±0.09	2.2±0.07	2.8±0.07	4±0.4	2±0.3	3.5±0.1
	w,t,l	11,0,0	10,0,1	7,2,2	2,1,8	0,0,11	2,1,8	7,2,2	6,0,5	5,0,6	1,0,10	7,2,2	4,0,7
	E_{BBC}	**0.68**±0.03	0.8±0.04	1.3±0.04	2.3±0.2	5.4±1	3.1±0.4	0.93±0.06	1.1±0.06	2.2±0.06	3.6±0.4	1.1±0.3	2.4±0.1
	w,t,l	11,0,0	10,0,1	6,0,5	4,0,7	0,0,11	2,0,9	9,0,2	7,1,3	5,0,6	1,0,10	7,1,3	3,0,8
w-l		88	70	34	-56	-77	-36	33	4	-28	-47	42	-27

Fig. 10 shows that all the four adaptive algorithms exhibit adaptive behaviors. On the 200-peak MPB problem, AMP/PSO, DynPopDE, and SAMO shows similar behaviors by which the number of populations gradually increases as the search goes on. Different from the above three algorithms, the number of populations of AMSO quickly converges at a certain level. Among the four adaptive algorithms, DynPopDE generates the largest number of populations in all cases.

2) Comparison on Problems With Varying Number of Peaks: A problem with a fixed number of peaks may be easy to solve. However, a problem with a varying number of peaks will challenge an algorithm's adaptability. Table XII and Fig. 11 present the errors of E_O and E_{BBC}, SR and PR, respectively, on problems with a varying number of peaks.

From Table XII, AMP/PSO and AMP/DE achieve significantly better performance than all the other algorithms in all the cases. The adaptive algorithms SAMO and AMSO also achieve relatively good results in comparison with the other nonadaptive algorithms. Among the nonadaptive algorithms, the number of populations in CPSOR is configured according to the number of peaks, which makes it behave as an adaptive algorithm. Therefore, CPSOR achieves good results.

Fig. 11 suggests that the four adaptive algorithms again show adaptive behaviors to the changing number of peaks,

where the number of populations is basically synchronous with the change of the number of peaks on the instance with Var1 (they also show such adaptive behavior in the case of Var3 if we observe the curves with Var3 closely). Among these four algorithms, AMP/PSO and SAMO show the best synchronization. DynPopDE again generates the largest number of populations, which makes it perform very poorly. Although all the adaptive algorithms show similar behaviors to AMP/PSO in terms of populations adaptation, they perform much worse than AMP/PSO regarding the errors E_O and E_{BBC}.

3) Effect of Varying the Change Frequency: Fig. 12 and Table XIII present the results of SR and PR, and the errors of E_O and E_{BBC}, respectively, for all the involved algorithms. Table XIII shows that AMP/PSO and AMP/DE achieve the best results in most cases. Increasing the change frequency means that algorithms will have more evaluations to locate and track optima before changes occur. Therefore, the performance of all the algorithms improves as the change frequency increases. It is interesting to observe that AMP/PSO achieves the greatest improvement in terms of PR among all the algorithms (Fig. 12), especially when $u > 5000$. Thanks to the adaptation mechanism, the AMP framework is able to make full use of the available evaluations to explore as many peaks as possible.

V. Conclusion

Identifying the correct number of populations is a key issue to apply MPMs to solving DOPs. In order to address this issue, this paper proposes an AMP framework. A database is used to record useful information for guiding the adjustment of the total number of populations. Learning from historical data makes the AMP framework robust to solve problems, and continuously providing feedback to the database helps the AMP framework achieve an effective adaptation. From the experimental results of the two instantiated algorithms with the AMP framework, several conclusions can be drawn. First, the AMP framework is able to adaptively adjust the number of populations according to the number of peaks in the fitness landscape. Second, the AMP framework achieves the best performance among all the peer algorithms in most test cases, especially with regard to the capability of tracking multiple peaks. Third, the AMP framework is capable of locating multiple peaks in both static and dynamic environments.

References

[1] S. Bird and X. Li, "Adaptively choosing niching parameters in a PSO," in *Proc. Genet. Evol. Comput. Conf.*, Seattle, WA, USA, 2006, pp. 3–10.

[2] S. Bird and X. Li, "Using regression to improve local convergence," in *Proc. IEEE Congr. Evol. Comput.*, Singapore, 2007, pp. 592–599.

[3] T. Blackwell, "Particle swarm optimization in dynamic environments," in *Evolutionary Computation in Dynamic and Uncertain Environments* (Studies in Computational Intelligence). Berlin, Germany: Springer, 2007, ch. 2, pp. 29–49.

[4] T. M. Blackwell and P. Bentley, "Don't push me! Collision-avoiding swarms," in *Proc. IEEE Congr. Evol. Comput.*, vol. 2. Honolulu, HI, USA, 2002, pp. 1691–1696.

[5] T. Blackwell and J. Branke, "Multi-swarm optimization in dynamic environments," in *Applications of Evolutionary Computation*, vol. 3005. Berlin, Germany: Springer, 2004, pp. 489–500.

[6] T. Blackwell and J. Branke, "Multiswarms, exclusion, and anti-convergence in dynamic environments," *IEEE Trans. Evol. Comput.*, vol. 10, no. 4, pp. 459–472, Aug. 2006.

[7] T. M. Blackwell and P. J. Bentley, "Dynamic search with charged swarms," in *Proc. Genet. Evol. Comput. Conf.*, New York, NY, USA, 2002, pp. 19–26.

[8] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, vol. 3. Washington, DC, USA, 1999, pp. 1875–1882.

[9] J. Branke, T. Kaussler, C. Smidt, and H. Schmeck, "A multi-population approach to dynamic optimization problems," in *Proc. 4th Int. Conf. Adapt. Comput. Design Manuf.*, Plymouth, U.K., 2000, pp. 299–307.

[10] J. Branke and H. Schmeck, "Designing evolutionary algorithms for dynamic optimization problems," in *Advances in Evolutionary Computing* (Natural Computing Series), A. Ghosh and S. Tsutsui, Eds. Berlin, Germany: Springer, 2003, pp. 239–262.

[11] I. G. del Amo, D. A. Pelta, and J. R. González, "Using heuristic rules to enhance a multiswarm PSO for dynamic environments," in *Proc. IEEE Congr. Evol. Comput.*, Barcelona, Spain, Jul. 2010, pp. 1–8.

[12] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: Harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[13] U. Halder, S. Das, and D. Maity, "A cluster-based differential evolution algorithm with external archive for optimization in dynamic environments," *IEEE Trans. Cybern.*, vol. 43, no. 3, pp. 881–897, Jun. 2013.

[14] M. Kamosi, A. B. Hashemi, and M. R. Meybodi, "A hibernating multi-swarm optimization algorithm for dynamic environments," in *Proc. World Congr. Nat. Biol. Inspir. Comput. (NaBIC)*, Fukuoka, Japan, 2010, pp. 363–369.

[15] M. R. Khouadjia, B. Sarasola, E. Alba, L. Jourdan, and E. Talbi, "Multi-environmental cooperative parallel metaheuristics for solving dynamic optimization problems," in *Proc. IEEE Int. Symp. Parallel Distrib. Process. Workshops PhD Forum (IPDPSW)*, Shanghai, China, May 2011, pp. 395–403.

[16] J. Lampinen and I. Zelinka, "On stagnation of the differential evolution algorithm," in *Proc. 6th Int. Conf. Soft Comput MENDEL*, Brno, Czech Republic, 2000, pp. 76–83.

[17] C. Li and S. Yang, "Fast multi-swarm optimization for dynamic optimization problems," in *Proc. 4th Int. Conf. Nat. Comput.*, vol. 7. Jinan, China, 2008, pp. 624–628.

[18] C. Li and S. Yang, "A clustering particle swarm optimizer for dynamic optimization," in *Proc. IEEE Congr. Evol. Comput.*, Trondheim, Norway, 2009, pp. 439–446.

[19] C. Li and S. Yang, "A general framework of multipopulation methods with clustering in undetectable dynamic environments," *IEEE Trans. Evol. Comput.*, vol. 16, no. 4, pp. 556–577, Aug. 2012.

[20] C. Li, S. Yang, and M. Yang, "An adaptive multi-swarm optimizer for dynamic optimization problems," *Evol. Comput.*, vol. 22, no. 4, pp. 559–594, 2014.

[21] L. Li and K. Tang, "History-based topological speciation for multimodal optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 1, pp. 136–150, Feb. 2015.

[22] R. I. Lung and D. Dumitrescu, "A collaborative model for tracking optima in dynamic environments," in *Proc. IEEE Congr. Evol. Comput.*, Singapore, 2007, pp. 564–567.

[23] R. I. Lung and D. Dumitrescu, "Evolutionary swarm cooperative optimization in dynamic environments," *Nat. Comput.*, vol. 9, no. 1, pp. 83–94, 2010.

[24] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge Univ. Press, 2008.

[25] R. Mendes and A. S. Mohais, "DynDE: A differential evolution for dynamic optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, Edinburgh, U.K., 2005, pp. 2808–2815.

[26] R. Mukherjee, G. R. Patra, R. Kundu, and S. Das, "Cluster-based differential evolution with crowding archive for niching in dynamic environments," *Inf. Sci.*, vol. 267, pp. 58–82, May 2014.

[27] T. T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm Evol. Comput.*, vol. 6, pp. 1–24, Oct. 2012.

[28] T. T. Nguyen, I. Jenkinson, and Z. Yang, "Solving dynamic optimisation problems by combining evolutionary algorithms with KD-tree," in *Proc. Int. Conf. Soft Comput. Pattern Recognit. (SoCPaR)*, Hanoi, Vietnam, Dec. 2013, pp. 247–252.

[29] D. Parrott and X. Li, "A particle swarm model for tracking multiple peaks in a dynamic environment using speciation," in *Proc. IEEE Congr. Evol. Comput.*, Portland, OR, USA, 2004, pp. 98–103.
[30] D. Parrott and X. Li, "Locating and tracking multiple dynamic optima by a particle swarm model using speciation," *IEEE Trans. Evol. Comput.*, vol. 10, no. 4, pp. 440–458, Aug. 2006.
[31] B. Y. Qu, P. N. Suganthan, and S. Das, "A distance-based locally informed particle swarm model for multimodal optimization," *IEEE Trans. Evol. Comput.*, vol. 17, no. 3, pp. 387–402, Jun. 2013.
[32] B. Y. Qu, P. N. Suganthan, and J. J. Liang, "Differential evolution with neighborhood mutation for multimodal optimization," *IEEE Trans. Evol. Comput.*, vol. 16, no. 5, pp. 601–614, Oct. 2012.
[33] I. Rezazadeh, M. R. Meybodi, and A. Naebid, "Adaptive particle swarm optimization algorithm for dynamic environments," in *Proc. Int. Conf. Adv. Swarm Intell.*, Chongqing, China, 2011, pp. 120–129.
[34] I. L. Schoeman and A. P. Engelbrecht, "Using vector operations to identify niches for particle swarm optimization," in *Proc. IEEE Conf. Cybern. Intell. Syst.*, vol. 1. Singapore, Dec. 2004, pp. 361–366.
[35] I. Schoeman and A. Engelbrecht, "Niching for dynamic environments using particle swarm optimization," in *Simulated Evolution and Learning* (LNCS 4247), T.-D. Wang *et al.*, Eds. Berlin, Germany: Springer, 2006, pp. 134–141.
[36] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE Congr. Evol. Comput.*, Anchorage, AK, USA, 1998, pp. 69–73.
[37] M. C. du Plessis and A. P. Engelbrecht, "Differential evolution for dynamic environments with unknown numbers of optima," *J. Global Optim.*, vol. 55, no. 1, pp. 73–99, 2013.
[38] R. Thomsen, "Multimodal optimization using crowding-based differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, vol. 2, Portland, OR, USA, 2004, pp. 1382–1389.
[39] A. M. Turky and S. Abdullah, "A multi-population harmony search algorithm with external archive for dynamic optimization problems," *Inf. Sci.*, vol. 272, pp. 84–95, Jul. 2014.
[40] N. J. Unger, B. M. Ombuki-Berman, and A. P. Engelbrecht, "Cooperative particle swarm optimization in dynamic environments," in *Proc. IEEE Symp. Swarm Intell. (SIS)*, Singapore, Apr. 2013, pp. 172–179.
[41] F. Van Den Bergh, "An analysis of particle swarm optimizers," Ph.D. dissertation, Dept. Comput. Sci., Univ. Pretoria, Pretoria, South Africa, 2002.
[42] H. Wang, N. Wang, and D. Wang, "Multi-swarm optimization algorithm for dynamic optimization problems using forking," in *Proc. Chin. Control Decis. Conf.*, Yantai, China, Jul. 2008, pp. 2415–2419.
[43] M. Yang, C. Li, Z. Cai, and J. Guan, "Differential evolution with auto-enhanced population diversity," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 302–315, Feb. 2015.
[44] S. Yang and C. Li, "A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments," *IEEE Trans. Evol. Comput.*, vol. 14, no. 6, pp. 959–974, Dec. 2010.
[45] D. Yazdani, B. Nasiri, A. Sepas-Moghaddam, M. Meybodi, and M. Akbarzadeh-Totonchi, "mNAFSA: A novel approach for optimization in dynamic environments with global changes," *Swarm Evol. Comput.*, vol. 18, pp. 38–53, Oct. 2014.
[46] D. Yazdani, B. Nasiri, A. Sepas-Moghaddam, and M. R. Meybodi, "A novel multi-swarm algorithm for optimization in dynamic environments based on particle swarm optimization," *Appl. Soft Comput.*, vol. 13, no. 4, pp. 2144–2158, 2013.

Changhe Li (M'12) received the B.Sc. and M.Sc. degrees from China University of Geosciences, Wuhan, China, in 2005 and 2008, respectively, and the Ph.D. degree from University of Leicester, Leicester, U.K., in 2011, all in computer science.

He has been an Associate Professor with the School of Computer Science, China University of Geosciences since 2012. His research interests include evolutionary algorithms with machine leaning, swarm intelligence, multimodal optimization, and dynamic optimization.

Dr. Li is the Vice Chair of the Task Force on Evolutionary Computation in Dynamic and Uncertain Environments.

Trung Thanh Nguyen received the B.Sc. degree in computer science from Vietnam National University, Hanoi, Vietnam, in 2000, and the M.Phil. and Ph.D. degrees in computing science from University of Birmingham, Birmingham, U.K., in 2007 and 2011, respectively.

He was a Research Fellow with Liverpool John Moores University (LJMU), Liverpool, U.K., and with University of Birmingham in 2011. He has been a Senior Lecturer of Optimisation and Simulation Modelling with LJMU since 2013, and a Reader of Operational Research since 2015. He is currently the Principal Investigator of five research grants in transport and logistics. He has published over 30 peer-reviewed papers. His research interests include operational research/dynamic optimization, with a particular application to logistics/transport problems.

Dr. Nguyen is the Chair of three leading conference tracks, was a member of Technical Programme Committees of over 20 leading conferences, an Editor of three books and two journals, and an Invited Speaker of various conferences and events.

Ming Yang received the B.Sc., M.Sc., and Ph.D. degrees in computer science from China University of Geosciences, Wuhan, China, in 2005, 2008, and 2012, respectively.

He is a Lecturer with China University of Geosciences. His research interests include evolutionary computation and dynamic optimization.

Michalis Mavrovouniotis (M'13) received the B.Sc. degree in computer science from University of Leicester, Leicester, U.K., in 2008; the M.Sc. degree in natural computation from University of Birmingham, Birmingham, U.K., in 2009; and the Ph.D. degree in computer science from University of Leicester in 2013.

He is a Research Associate with the Centre for Computational Intelligence, De Montfort University, Leicester. His research interests include evolutionary computation, swarm intelligence, memetic computing, combinatorial optimization problems, optimization problems with dynamic and uncertain environments, and relevant real-world applications.

Shengxiang Yang (M'00–SM'14) received the B.Sc. and M.Sc. degrees in automatic control and the Ph.D. degree in systems engineering from Northeastern University, Shenyang, China, in 1993, 1996, and 1999, respectively.

He is a Professor of Computational Intelligence and the Director of the Centre for Computational Intelligence, School of Computer Science and Informatics, De Montfort University, Leicester, U.K. He has over 190 publications. His research interests include evolutionary and genetic algorithms, swarm intelligence, computational intelligence in dynamic and uncertain environments, artificial neural networks for scheduling, and relevant real-world applications.

Prof. Yang is the Chair of the Task Force on Evolutionary Computation in Dynamic and Uncertain Environments, under the Evolutionary Computation Technical Committee of the IEEE Computational Intelligence Society and the Founding Chair of the Task Force on Intelligent Network Systems, under the Intelligent Systems Applications Technical Committee of the IEEE Computational Intelligence Society.